York University
CSE 2001 – Unit 4.0 Context Free Grammars and Parsers and Context Sensitive Grammars

Instructor: Jeff Edmonds

Read Jeff's notes. Read the book. Go to class. Ask lots of question. Study the slides. Work hard on solving these questions on your own. Talk to your friends about it. Talk to Jeff about it. Only after this should you read the posted solutions. Study the solutions. Understand the solutions. Memorize the solutions. The questions on the tests will be different. But the answers will be surprisingly close.

1. Consider the following grammar.

$$\begin{aligned}
\text{exp} &\Rightarrow \text{term}\\
&\Rightarrow \text{term} + \text{exp}\\
&\Rightarrow \text{term - exp}\\
\text{term} &\Rightarrow \text{fact}\\
&\Rightarrow \text{fact * term}\\
\text{fact} &\Rightarrow \text{int}\\
&\Rightarrow \text{(exp)}
\end{aligned}$$

   (a) Give a derivation/parsing tree of the following expressions. (You may use the single letters e, t, and f for exp, term, and fact.

      i. $s =$ "9+3*2".
      ii. $s =$ "(9+3)*2".
      iii. $s =$ "9-3-2".
      iv. $s =$ "(((1) * 2 + 3) * 5 * 6 + 7)".

   (b) Syntactics vs Semantics

      i. Syntactically, does this grammar generate reasonable expressions?
      ii. Semantically (meaning), does it make any mistakes, i.e. if you evaluated the above expressions using your parsings, would you get the correct answers?

2. Recall that the regular languages are closed under complement, union, and intersection. Clearly context free languages are closed under union because if L_1 is generated with the grammar $S_1 \Rightarrow ....$ and L_2 with $S_2 \Rightarrow ....$, then $L = L_1 \cup L_2$ is generated with the grammar $S \Rightarrow S_1 \mid S_2$ together with the other rules. We will now prove that context free languages are NOT closed under intersection or under complement. Specifically, we define two languages $a^nb^nc^*$ and $a^*b^nc^n$ that are context free, however, $a^nb^nc^n = a^nb^nc^* \cap a^*b^nc^n$ is not context free. Similarly, we define language $L_?$ that is context free, however, $\overline{L_?}$ is not context free. At the end, we give a simpler example that was inspired by the fact that $\overline{L_1 \cap L_2} = \overline{L_1} \cup \overline{L_2}$ is context free but $L_1 \cap L_2$ is not.

   - $L_{=*} = a^nb^nc^*$.
     Clearly this is context free.
   - $L_{*=} = a^*b^nc^n$.
     Clearly this is context free.
   - $L_{==} = a^nb^nc^n$.
     In class we argued that this language cannot be done by a context free grammar.
   - We will prove that $L_{==} = L_{=*} \cap L_{*=}$.
     This proves that context free languages are NOT closed under intersection.
   - $L_= = \{w_1\#w_2 \mid w_1, w_2 \in \{0,1\}^* \text{ and } w_1 = w_2\}$.
     In class we argued that this language cannot be done by a context free grammar.

- $L_{\neq} = \{w_1\#w_2 \mid w_1, w_2 \in \{0,1\}^* \text{ and } w_1 \neq w_2\}$.
  This question here will develop a context free grammar for this.
- $L_{\#\#} = \{w \in \{0,1,\#\}^* \mid \text{the number of } \# \text{ in } w \text{ is not one } \}$.
  Clearly this is context free.
- $L_? = L_{\neq} \cup L_{\#\#}$.
  Clearly this is context free.
- Note the full universe $\{0,1,\#\}^*$ is the disjoint union of $L_=$, $L_{\neq}$, and $L_{\#\#}$.
  Hence, $\overline{L_=} = L_{\neq} \cup L_{\#\#} = L_?$ and $\overline{L_?} = L_=$.
  Note $L_?$ is context free and $\overline{L_?} = L_=$ is not.
  This proves that context free languages are NOT closed under intersection.

We now fill in the details.

(a) How did we argue in class that $a^nb^nc^n$ cannot be done by a context free grammar.

(b) Argue that $a^nb^nc^n = a^nb^nc^* \cap a^*b^nc^n$.

(c) The build a grammar for $\{0,1\}^*$.

- Answer 1:
  $S \Rightarrow CS \mid \epsilon$
  $C \Rightarrow 0 \mid 1$
- Answer 2: I will now give the same grammar again but with different names for the non-terminals. In order to generate the languages $L_{\{0,1\}^*} = \{0,1\}^*$ and $L_{\{0,1\}} = \{0,1\}$, I will name the start non-terminal with $S_{\{0,1\}^*}$ and $S_{\{0,1\}}$.
  $S_{\{0,1\}^*} \Rightarrow S_{\{0,1\}}\ S_{\{0,1\}^*} \mid \epsilon$
  $S_{\{0,1\}} \Rightarrow 0 \mid 1$

(d) Give a grammar for $L_{=*} = a^nb^nc^*$ starting with non-terminal $S_{=*}$.

(e) How did we argue in class that
$L_= = \{w_1\#w_2 \mid w_1, w_2 \in \{0,1\}^* \text{ and } w_1 = w_2\}$.
cannot be done by a context free grammar.

(f) The build a grammar for $L_* = \{w_1\#w_2 \mid w_1, w_2 \in \{0,1\}^*\}$ starting with the non-terminal S_*.

(g) We will now develop a context free grammar for
$L_{\neq} = \{w_1\#w_2 \mid w_1, w_2 \in \{0,1\}^* \text{ and } w_1 \neq w_2\}$.
Note that if two strings w_1 and w_2 are different, then either these strings have different lengths or there is some $n \geq 0$ such that the $n{+}1^{st}$ bit of w_1 is different than that of w_2. Note we use this strange indexing because there are n bits before the $n{+}1^{st}$ bit. For example, $101\#1011 \in L_{\neq}$, because the strings 101 and 1011 have different lengths. On the other hand, $101\#111 \in L_{\neq}$, because the strings 101 and 111 are different in the 2^{nd} bit, i.e. the $n{+}1^{st}$ bit when $n = 1$.
With this motivation, define the following four languages.

- $L_{n>m} = \{w_1\#w_2 \mid w_1, w_2 \in \{0,1\}^* \text{ and } |w_1| > |w_2|\}$.
- $L_{n<m} = \{w_1\#w_2 \mid w_1, w_2 \in \{0,1\}^* \text{ and } |w_1| < |w_2|\}$.
- $L_{0\neq1} = \{w_1\#w_2 \mid w_1, w_2 \in \{0,1\}^*, \text{ some } n{+}1^{st} \text{ bit of } w_1 \text{ is } 0, \text{ and of } w_2 \text{ is } 1\ \}$.
- $L_{1\neq0} = \{w_1\#w_2 \mid w_1, w_2 \in \{0,1\}^*, \text{ some } n{+}1^{st} \text{ bit of } w_1 \text{ is } 1, \text{ and of } w_2 \text{ is } 0\ \}$.

Suppose you already have grammars for each of these languages starting with the non-terminal named $S_{n>m}$, $S_{n<m}$, $S_{0\neq1}$, and $S_{1\neq0}$. Give a grammar for $L_{\neq}$ starting with the non-terminal $S_{\neq}$.

(h) Recall that a grammar is *ambiguous* if there are more than one parsings for the same string. Argue whether or not this grammar for $L_{\neq}$ is ambiguous.

(i) Give grammars for the following languages:
$L_{n=m} = \{0,1\}^n\#\{0,1\}^n = \{w_1\#w_2 \mid w_1, w_2 \in \{0,1\}^* \text{ and } |w_1| = |w_2|\}$ starting with $S_{n=m}$.

(j) $L_{n>m} = \{\{0,1\}^n\#\{0,1\}^m \mid n > m\} = \{w_1\#w_2 \mid w_1, w_2 \in \{0,1\}^* \text{ and } |w_1| > |w_2|\}$ starting with $S_{n>m}$.

(k) Argue that the following two languages are the same.
$L_{0\neq 1} = \{w_1\#w_2 \mid w_1, w_2 \in \{0,1\}^*$, some $n+1^{st}$ bit of w_1 is 0, and of w_2 is 1 $\}$
and $\{0,1\}^n0\{0,1\}^*\#\{0,1\}^n1\{0,1\}^*$.

(l) $L_{0\neq 1} = \{0,1\}^n0\{0,1\}^*\#\{0,1\}^n1\{0,1\}^*$ starting with $S_{0\neq 1}$.
Hint: Which blocks need to be linked and hence must get spewed together?

(m) This completes the grammar for $L_{\neq}$. I find it surprising that this language has a grammar.

(n) $L_{\#\#} = \{w \in \{0,1,\#\}^* \mid$ the number of # in w is not one $\}$ starting with $S_{\#\#}$.

(o) $L_? = L_{\neq} \cup L_{\#\#}$ starting from $S_?$.

(p) Here is a second example of a language $L_?$ that is context free, however, $\overline{L_?}$ is not context free. This example was inspired by the fact that $L_1 \cap L_2$ is not context free but $\overline{L_1 \cap L_2} = \overline{L_1} \cup \overline{L_2}$ is.

- $L_{==} = \{0^\ell\#0^m\#0^n \mid \ell = m \text{ and } m = n\} = 0^n\#0^n\#0^n$.
  In class we argued that this language cannot be done by a context free grammar.
- $L_{\neq\neq} = \{0^\ell\#0^m\#0^n \mid \ell \neq m \text{ or } m \neq n\}$.
  Note that the complement of ($\ell = m$ and $m = n$) is (m or $m \neq n$). It is not too hard to see that this is context free.
- $L_{\#\#\#} = \{w \in \{0,\#\}^* \mid$ the number of # in w is not two $\}$.
  Clearly this is context free.
- $L_? = L_{\neq} \cup L_{\#\#}$.
  Clearly this is context free.
- Note the full universe $\{0,\#\}^*$ is the disjoint union of $L_{==}$, $L_{\neq\neq}$, and $L_{\#\#\#}$.
  Hence, $\overline{L_{==}} = L_{\neq\neq} \cup L_{\#\#\#} = L_?$ and $\overline{L_?} = L_{==}$.
  Note $L_?$ is context free and $\overline{L_?} = L_{==}$ is not.
  This proves that context free languages are NOT closed under intersection.

Fill in the details.

3. Parsing

Look Ahead One: A grammar is said to be *look ahead one* if, given any two rules for the same non-terminal, the first place that the rules differ is a difference in a terminal. (Equivalently the rules can be views as paths down a tree.) This feature allows our parsing algorithm to look only at the next token in order to decide what to do next. Thus the algorithm runs in linear time. An example of a good set of rules would be:

A ⇒ B 'u' C 'w' E
A ⇒ B 'u' C 'x' F
A ⇒ B 'u' C
A ⇒ B 'v' G H

(Actually, even this grammar could also be problematic if when $s = bbbucccweee$, B could either be parsed as bbb or as $bbbu$. Having B *eat* the $'u'$ would be a problem.)
An example of a bad set of rules would be:

A ⇒ B C
A ⇒ D E

With such a grammar, you would not know whether to start parsing the string as a B or a D. If you made the wrong choice, you would have to back up and repeat the process.

Consider a grammar G which includes the four look ahead rules for A given above. Give the code for $GetA\,(s,i)$ that is similar to that for $GetExp\,(s,i)$. We can assume that it can be parsed, so do not bother with error detection. HINT: The code should contain NO loops.

4. Context Sensitive Grammar: There are no context free grammars for generating the language $\alpha\#\alpha$ where $\alpha \in \{0,1\}^*$. (The proof is an ugly pumping lemma thing, which you don't have to do.) Give a context sensitive grammar for generating this language.

   A loop invariant stating what string that grammar now has is REQUIRED. Give meaning and explanation to your non-terminals.

   The following are hints about two different ways to do it. Think about both, but only hand in one.

   (a) Start with $S \rightarrow S'{<}$
   $S' \rightarrow 0S'0 \mid 1S'1 \mid \#{>}Q$
   This produces the string $\alpha\#{>}Q\alpha^R{<}$
   Then move the head Q back and forth to reverse the order of α^R giving α.

   (b) Start with $S \rightarrow S'{<}$
   $S' \rightarrow 0S'C_0 \mid 1S'C_1 \mid \#$
   This produces the string $\alpha\#\beta{<}$
   where $\beta = \alpha^R$, except 0 is replaced by C_0 and 1 by C_1.
   Then have the C_0 and the C_1 move on their own and convert when in place to 0 and 1.